Exerciser

clemperador

Compilation and execution:

To compile, use the command:
`gcc exer.c -o ex -lm`. Alternatively, an already-compiled file`ex` is included.

To execute the compiled file, use the command:
`./ex`

The program will then ask if the user wishes to generate arithmetical expressions or quadratic equations.

Implementation choices:

Arithmetical Expressions Generator:

The generation of arithmetical expressions begins at `char* expression` who returns a string with an arithmetical expression. It takes `int n, d, lvl` as parameters.

- `n` is the number of sub-calculations per problem.
- `d` is the max digits per number.
- `lvl` is the operation's 'difficulty'. 0 means the problem's and sub-problem's results will be positive. 1 means the opposite.

These parameters are asked as input by the interactive function `void arith_interac()`.

The function generates a random number that, depending on its value, assigns an operator `(+, -, *, /)` to concatenate to the final expression. This is also used to link sub-operations between each other with operators. Parentheses are also implemented, that can be added to the expression randomly. They are properly closed at the end of the program's execution.

If `lvl == 1`, operations are generated until one has each of its sub-operations bringing a positive result. Divisions (and thus, all final expressions) always have integer results.

The function `void savefile()`, taking as parameters `n, d, lvl`, and `k` (also asked as input), that designates the number of expressions to save, saves the expressions in the file `expressions.txt`.

To evaluate expressions and save their results, and to test sub-operatoins, we use the NPI calculator in the folder `infixpostfix`. Its usage is explained (in french) on a README file within that folder. To use it here, we use the `system()` function.

`system("cd infixpostfix && echo "[`**EXPRESSION**`]> ../temp.txt")`

The result is saved on a temporal file `temp.txt` if the evaluation is temporal, or in `expanswers.txt` if the results are definitive. Spaces `(" ")` in expressions are always deleted before evaluation, to avoid errors with the program `infixpostfix`.

Quadratic Equations Generator.

The Generation of Quadratic Equations follows a logic similar to that of arithmetical expressions. Three integers `a, b, et c`, corresponding to the coefficients of $ax^2 + bx + c$, are randomly generated, also with a number of digits entered as parameter. The results in `expanswers.txt` will be the polynomial's roots. We want integer, real roots. We use the following property:

A polynomial $ax^2 + bx + c$ allows integer roots if and only if
1. $b^2 - 4ac$ is a perfect square or integer.
2. a divises b and c

We generate then the asked number of polynomials, saving them only if the conditions are well followed. A function `isperfsquare(int d)` is used to verify the

discriminator. Both solutions are then computed with the formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

If both solutions are equal, ($b^2 - 4ac = 0$), the double root is considered as one.
The roots of each polynomial saved in `expressions.txt` are saved on its corresponding line in `expanswers.txt`

Print:

The exerciser can also print the generated text files, with the `system("cat expressions.txt)` command.

Overture:
Possible Improvements:

Many of the functions that check the basic conditions of the expressions can take a very long time if a large number of digits or sub-computations are asked. Optimizations to these functions could fix this. The usage of more sophisticated data structures such as a binary tree is an example of a more optimized implementation.

Possible future features:

The exerciser could also allow the user to solve interactively the generated expressions, comparing the inputted answers with those in `expanswers.txt`.